Software Lifecycle Themes for JPL Mission Data System (MDS)

Anne Elson
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099

Abstract

Today there are many small deep space missions in progress or in conception at the Jet Propulsion Laboratory. These missions have short lead times and small budgets while still pursuing ambitious science and technology goals. The short development times, the small funding profiles and the overlapping schedules of these new missions preclude the intensive, one-of-a-kind software development, maintenance, and operations efforts that were possible during the era of the big missions like Galileo and Cassini. How the laboratory develops, maintains and operates mission software in this new environment of multiple concurrent better, faster, and cheaper (BFC) missions will be crucial to the success of these new missions. The Mission Data System Project (MDS) team is developing core mission data system software for a group of the new BFC missions. As part of this effort the MDS team is piloting the use of a number of state-of-the art software development, maintenance and operations approaches and tools. This paper describes MDS software lifecycle strategies and tools and the ways in which these differ from past software development, maintenance and operations efforts at the laboratory. Additionally this paper discusses how the MDS software lifecycle approach can contribute to the success of the new BFC missions.

Extended Abstract

Mission Data System Overview

In April of 1998 the Jet Propulsion Laboratory initiated the Mission Data System Project (MDS). This project is chartered with rethinking the entire mission software lifecycle. MDS has proposed a unified flight, ground and test data system architecture for space missions. This architecture is characterized by a number of architectural themes. These themes and how they will contribute to the success of better, faster, cheaper missions are discussed in a related conference paper "Software Architectural Themes in the Mission Data System". The MDS software system will include a core set of standard flight, ground and test software capabilities that mission customers typically need. Additionally it will contain a set of partially fleshed out software capabilities (frameworks) and some example implementations of these capabilities. These frameworks and example implementations will be for capabilities that are more likely to be mission unique or have mission unique aspects. Customer missions will build upon the MDS core system by using or adapting MDS reference examples and/or by filling in and building upon the relevant frameworks to meet their own mission specific needs.

The MDS system will provide mission software capabilities that are equivalent to mission software capabilities of recent JPL deep space missions. Additionally MDS team is incorporating a set of new and innovative design concepts into MDS software that will result in a set of new, and/or expanded mission software capabilities for current and for future mission customers. One of the team's goals is to produce a software product that enables spacecraft for deep space missions to be both more autonomous and easier to operate. Another is to provide a unified flight ground software system that facilitiates movement of software capabilities between the ground system and the spacecraft. Location of software capabilities such as spacecraft trajectory correction determination may depend upon the needs and constraints of a particular mission and could migrate from the ground to the spacecraft as a mission's needs changed over time. Yet another MDS goal is to produce software that is both reusable and adaptable for many different missions.

MDS Software Lifecycle Approach Overview

A lifecycle model provides a set of development guidelines to the developers. It identifies a set of development phases and the work products (artifacts) to be produced in each. The model helps to bring structure and standardization (predictability) to an activity that often appears to be chaotic and unpredictable. The MDS software lifecycle can be viewed as a set nested loops: one outer loop and an associated set of inner loops. The MDS project traverses one cycle of its outer loop and multiple cycles of

its inner loops to produce MDS flight, ground and test software for a single mission customer. The project's OOA/OOD consultant, Bruce Douglass, promotes a spiral development lifecycle model with iterative proto-typing in his forthcoming book "Rapid Object-Oriented Process for Embedded Systems". In the Douglass model developers repeat a set of major lifecycle phases multiple times. During the earliest iterations of this lifecycle developers implement a complete but thin version of the entire system. All system interfaces are defined and implemented but internals of many of software components attached to these interfaces are missing or only sketchily implemented. During each subsequent iteration of the lifecycle development teams increase the capabilities of the components they own in the evolving software system. Work products (artifacts) grow in their completeness and quality until all agreed upon system requirements and constraints are achieved (or re-negotiated). The MDS lifecycle model includes iterative proto-typing in its inner loop cycles.

In the MDS lifecycle the outer loop of the lifecycle maps to the management and software system engineering activities that the MDS team will perform to produce an MDS system for a single customer. The outer loop divides into 4 phases: feasibility, elaboration, construction and transition. The MDS outer loop is primarily incremental but phases overlap. The outer loop includes requirements and systems analysis, system design activities, system test and validation activities, and system maintenance activities.

MDS inner loop activities and phases closely follow Douglass' software lifecycle model. For each circuit of the MDS outer loop MDS development teams will complete multiple iterations (cycles) of a set of associated inner loops. Each MDS inner loop represents one software domain within the current MDS development effort. An inner loop is divided into 3 phases of software development: analysis and design, implementation, and evaluation and test. An MDS development team responsible for inner loop activities will traverse an inner loop cycle multiple times during one cycle of the MDS outer loop. Inner loop cycles may run asynchronously to one another part of the time. During inner loop iterations there will also be some planned, periodic alignments of inner loop completions across multiple software domains (sometimes referred to as synchronization points). These will occur when a capability crosses domains and requires coordination across teams or when system test schedules require the delivery of new integrated software capabilities to the project for system wide integration and test.

The MDS team is using object oriented analysis and design techniques. The team uses UML (Unified Modeling Language) to capture their analysis and design decisions within an OOA/OOD case tool. There are several OOA/OOD software case tools available commercially. The MDS team has chosen to use a tool called Rhapsody (this tool is produced by I-Logix). Using UML MDS team captures their object oriented analysis and design decisions as a series of models within their case tool. The team's OOA/OOD consultant, Bruce Douglass, defines a system model as "an organized, internally-consistent set of abstractions that collaborate to achieve a system description at a desired level of detail and maturity." Thoughout an MDS system lifecycle the MDS team develops and/or refines and updates system models. The MDS software team will develop analysis models, design models, translation (source code) models and testing models. Each model is another view of the underlying system and is not independent of the other views.

The remainder of this paper will discuss a number of software lifecycle themes that are a result of the MDS team's use of OOA/OOD methodologies. It will discuss how the tool and OOA/OOD methodologies help the team to verify and maintain consistency between work products as the team moves through one iteration of an MDS lifecycle. It will discuss how OOA/OOD in conjunction with the Rhapsody case tool enables the team to produce executable models of the underlying system throughout its development. It will discuss how the MDS lifecycle approach de-couples mission software development from flight hardware development, and promotes the treatment of software as a single system within the mission. It will discuss how this approach can contribute to the success of BFC missions at the lab. It will discuss how the MDS lifecycle approach promotes software reuse and adaptation. It will discuss how the OOA/OOD approach in conjunction with good metrics collection can reduce development risk by making the software lifecycle process more predictable and consistent. It will discuss how OOA/OOD methodologies in conjunction with a case tool that auto-generates code can considerably shorten mission software development times once the MDS team has a library of re-usable analysis and design models in place.